# DD2420 - Tutorial 4

Magnus Pierrau

February 2020

## 1 Question 1

Figures 1 - 3 show the GraphSLAM formulation as a Bayesian Belief Network, Factor Graph and Markov Random Field respectively. Here $x_t$ is the robot pose at time step $t$, $z_i$ is the $i$th measure, $m_j$ the $j$th measured landmark and $u_t$ the control signal between $x_{t-1}$ and $x_t$.

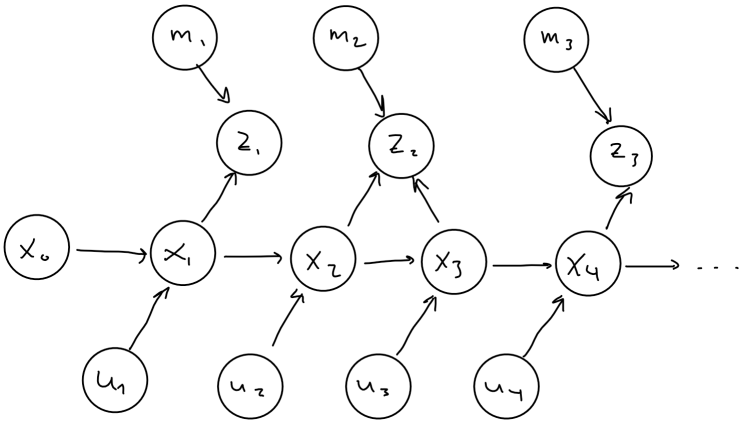


Figure 1: The GraphSLAM model illustrated as a Bayesian Belief Network.

## 2 Question 2

What assumptions are made in equation (1)?

$$p\left(x_{0:N}, m_{1:M}, z_{1:k}, u_{1:N}\right) = p\left(x_0\right) \prod_{i=1}^{N} p\left(x_i | x_{i-1}, u_i\right) \prod_{k=1}^{K} p\left(z_k | x_{p(k)}, m_{f(k)}\right) \quad (1)$$

Here we are assuming that each robot pose $x_i$ is only dependent on the previous pose as well as the control signal for the current movement. We are also

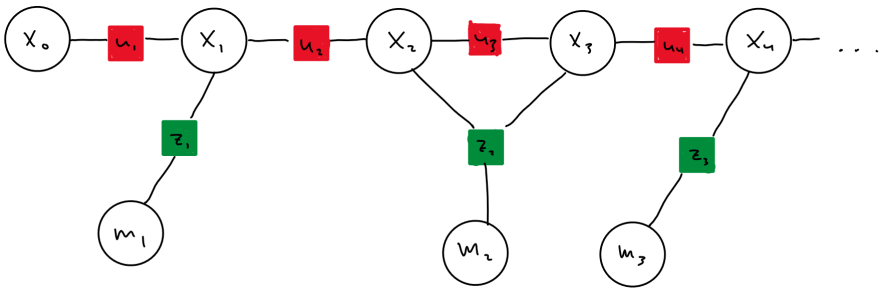

Figure 2: The GraphSLAM model illustrated as a Factor Graph.

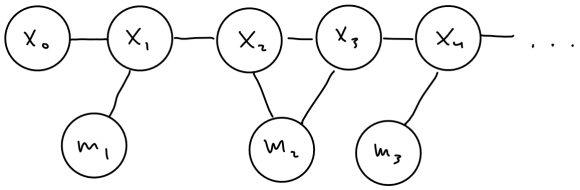

Figure 3: The GraphSLAM model illustrated as a Markov Random Field.

assuming that the measurement $z_k$ is only depending on the poses $x_{p(k)}$ from which the measurement was observed, as well as the landmarks $m_{f(k)}$ which were measured. We are also assuming that the initial pose $x_0$ is completely independent. Furthermore we are assuming that all $x_i$ are identically distributed, as well as all $z_k$.

# 3 Question 3

When the robot observes a previously seen landmark (for example if going back the path it came from or observes something identical to before), say $m_j$ at pose $x_{t2}$, and previously seen at $x_{t1}$ the model will create a new entry, $\mathbf{\Omega}_{j,t2}$ in the information matrix that is far from the diagonal (in the sub-matrix $\mathbf{\Omega}_{[0,j]}$). It introduces a non-zero element off the diagonal, creating a dependency between the two $x_{t1}, x_{t2} \in p(k)$. Previously $x_{t1}$ and $x_{t2}$ were only linked through the control sequence $u_{t1}, u_{t1+1}..., u_{t2}$. This makes the matrix non-band diagonal and does not allow us to reorder the equations in the system. We can thus no longer solve the system in linear time, having to rely on optimization techniques such as gradient descent.

Sometimes this observation is erroneous, creating a false loop-closure, which can lead to divergence of the algorithm, as seen in some of the practical examples.

# 4 Question 4

Show that linearizing the motion equation for one pose factor gives rise to the following terms to be added to $\mathbf{\Omega}$ and $\boldsymbol{\xi}$ at the appropriate places:

$$\begin{aligned}
\Delta\mathbf{\Omega}_{i-1,i-1} &= G_i^T R_i^{-1} G_i && (2)\\
\Delta\mathbf{\Omega}_{i,i} &= R_i^{-1} && (3)\\
\Delta\mathbf{\Omega}_{i,-1} &= -R_i^{-1} G_i && (4)\\
\Delta\boldsymbol{\xi}_{i-1} &= G_i^T R_i^{-1}(G_i\bar{\mu}_{i-1} - g(\bar{\mu}_{i-1}, u_i)) && (5)\\
\Delta\boldsymbol{\xi}_i &= R_i^{-1}(g(\bar{\mu}_{i-1}, u_i) + G_i\bar{\mu}_{i-1}) && (6)
\end{aligned}$$

Here $G$ is the Jacobian of the movement function $g$ with respect to $x_{i-1}$. This term appears when we linearize the system by doing a first order Taylor expansion of $g$ around $\bar{\mu}_{i-1}$. Here $\bar{\mu}_{i-1}$ is the current best estimate of $x_{i-1}$, acquired in the previous step of the algorithm. In particular we are performing the Taylor expansion in order to make the log likelihood linear in terms of $x_i$, not in terms of $g$.

We introduce a prior factor for the initial step, $p(x_0) = \eta \exp\left\{-\frac{1}{2} x_0^T \Omega_0 x_0\right\}$, giving the system a global anchoring constraint to the origin ($x_{00} = 0, x_{01} = 0, \theta_0 = 0$) (here the $i$ index of $x_{ti}$ indicates the dimension of $x$ and $\theta_t$ is the angular direction). Without this term we can only infer relative information on the poses, and no absolute estimates of the position and map of the system.

The log likelihood is then given by

$$\begin{aligned}
&-\ln p(x_{0:i} \mid z_{1:k}, m_{1:j}, u_{0:i})\\
&\qquad = C + \ln p(x_0) + \ln(x_{1:N} \mid Z_{1:K}, m_{1:M}, u_{1:N}) && (7)\\
&\qquad = C - \ln p(x_0) - \sum_{i=1}^{N} \ln p(x_i \mid x_{i-1}, u_i) && (8)\\
&\qquad\quad - \sum_{k=1}^{K} p(z_k \mid x_k, m_j) && (9)\\
& && (10)
\end{aligned}$$

We now have that, since $g$and $h$ have Gaussian error terms:

$$\begin{aligned}
p(x_i \mid x_{i-1}, u_i) &\sim \mathcal{N}(x_i - g(x_{i-1}, u_i), R_i)\\
p(z_k \mid x_i, m_j) &\sim \mathcal{N}(z_k - h(x_i, m_j), Q_i). && (11)\\
& && (12)
\end{aligned}$$

This gives us a (negative) log likelihood, $\mathcal{L}$, on the form

$$-\mathcal{L} = \quad C_1 + \frac{1}{2}\Bigg\{ x_0^T \Omega_0 x_0 + \sum_{i=1}^{N} \left(x_i - g(x_{i-1}, u_i)\right)^T R_i^{-1} \left(x_i - g(x_{i-1}, u_i)\right) \quad (13)$$

$$+ \sum_{k=1}^{K} \left(z_k - h(x_i, m_j)\right)^T Q_i^{-1} \left(z_k - h(x_i, m_j)\right) \Bigg\} \quad (14)$$

This equation is linear in $g$ and $h$, but we want linearity in $x_i$ and $x_{i-1}$. We thus make a Taylor expansion around the current estimate of $x_{i-1}$, $\bar{\mu}_{i-1}$. We get that

$$g(x_{i-1}, u_i) \approx g(u_i, \bar{\mu}_{i-1}) + G_i(x_{i-1} - \bar{\mu}_{i-1}) \quad (15)$$

$$(16)$$

In order to get the approximate update equations for the measurement function $h$ we would have to do the same expansion for $h$, but since only the linearization of the movement function, and not measurement function, is requested we omit these calculations.

Insertion of the approximation of $g$ into the log likelihood yields

$$-\mathcal{L} = C_2 + \frac{1}{2}\Bigg\{ x_0^T \Omega_0 x_0 \quad (17)$$

$$+ \sum_{i=1}^{N} \left(x_i - [g(\bar{\mu}_{i-1}, u_i) + G_i(x_{i-1} - \bar{\mu}_{i-1})]\right)^T R_i^{-1} \quad (18)$$

$$\left(x_i - [g(\bar{\mu}_{i-1}, u_i) + G_i(x_{i-1} - \bar{\mu}_{i-1})]\right) \quad (19)$$

$$+ \sum_{k=1}^{K} \left(z_k - h(x_i, m_j)\right)^T Q_i^{-1} \left(z_k - h(x_i, m_j)\right) \Bigg\} \quad (20)$$

Focusing now on the terms containing $x_i$ and $x_{i-1}$ we find that we can group these after whether they are linear or quadratic in these variables. It is also helpful to absorb constant terms into $C_3$.

$$
\begin{aligned}
-\mathcal{L} =& C_3 + \frac{1}{2}\Bigg\{ x_0^T \Omega_0 x_0 \\
&+ \sum_{i=1}^{N} x_i^T R_i^{-1} x_i - x_i^T R_i^{-1} G_i x_{i-1} && (21)\\
&- x_{i-1}^T G_i R_i^{-1} x_i + x_{i-1}^T G_i R_i^{-1} G_i x_{i-1} && (22)\\
&- x_i^T R_i^{-1} [g(\bar{\mu}_{i-1}, u_i) + G_i \bar{\mu}_{i-1}] && (23)\\
&+ x_{i-1}^T G_i R_i^{-1} [g(\bar{\mu}_{i-1}, u_i) + G_i \bar{\mu}_{i-1}] && (24)\\
&+ \sum_{k=1}^{K} \dots \Bigg\}
\end{aligned}
$$

We have now gathered all the quadratic terms in rows (21) and (22), and all linear terms in rows (23) and (24) (the sum over $k$ is not relevant and is therefore omitted for brevity). With some helpful notation for concatenating $x_i$ and $x_{i-1}$ as $x_{i-1:i}^T = (x_i \; x_{i-1})$, we have that the expression can be rewritten as

$$
\begin{aligned}
-\mathcal{L} =& C_3 + \frac{1}{2}\Bigg\{ x_0^T \Omega_0 x_0 && (25)\\
&+ \sum_{i=1}^{N} x_{i-1:i}^T \begin{pmatrix} 1 \\ -G_i \end{pmatrix} R_i^{-1} (1 \;\; -G_i) x_{i-1:i} && (26)\\
&- x_{i-1:i}^T \begin{pmatrix} 1 \\ -G_i \end{pmatrix} R_i^{-1} \left[ g(\bar{\mu}_{i-1}, u_i) + G_i \bar{\mu}_{i-1} \right] && (27)\\
&+ \sum_{k=1}^{K} \dots \Bigg\} && (28)
\end{aligned}
$$

By matching linear and quadratic expressions with equation (7I from the instructions, we can discern what each entry of $\Omega$ and $\xi$ will be by choosing the appropriate index in the vectors $x_{i-1:i}$, $\begin{pmatrix} 1 \\ -G_i \end{pmatrix}$ and its transpose respectively. Doing so for the quadratic terms and linear terms gives us the sought for equations (2)-(6) for $\boldsymbol{\Delta\Omega}$ and $\boldsymbol{\Delta\xi}$ respectively.

# 5 Question 5

We are trying to estimate the robot poses, and thus want the algorithm to converge when our estimates no longer change. We can therefore calculate the difference between the current and previous estimate for all poses and stop when this sum falls below some threshold. An alternative would be to consider the relative change and stop when this falls below some percentage. However, there are issues with this convergence criteria.

# 6 Question 6

The number of iterations required to converge will depend on the initial estimate $\bar{\mu}_0$ of $x_0$. Of course also the controls with affect the convergence as it creates a different system of equations to solve. Whether we have observed cycles or not will also affect the convergence, as the two scenarios require different means for solving the system. Of course the nature of the world matters, the number of landmarks and their relative position, as well as the amount of noise in the measurement and movement functions. Additionally we will need to consider the length of the path (number of time steps), as the errors accumulate over time, increasing the difficulty of inference.

# 7 Question 7

We might get caught in a loop where the error doesn't decrease, but instead fluctuates between two values. Also, when the estimates become too erroneous the algorithm fails to converge. This is due to the robot believing it is somewhere else and possibly creates false loops - i.e. connections between poses and landmarks that are not actually present.

# 8 Question 8

Characterize each of the five data sets:

1. What properties make them easier or harder?
2. Do you see evidence of local minima or very weak minima?
3. How does density of the map and the sensor range effect the convergence?
4. How does the amount of noise effect results?
5. What about the starting point?
6. When is the linearize system a reasonable approximation or not?
7. What happens when the robot closes a loop returning to see a landmark from the beginning?

Given maps:

## 8.1 `map_o3`

This is a somewhat larger and denser map, but with less noise in the measurements.

A denser map means that more landmarks are observed from more poses, linking multiple poses together by their joint observation of said landmark. As

we reduce $\mathbf{\Omega}$ and $\boldsymbol{\xi}$ this allows information to flow between said poses, improving the estimates, but simultaneously increasing the difficulty of solving the system.

As this map is larger, it requires more poses to complete a circulation of the map. We find that the estimates degrade over time due to accumulating measurement errors. One effective way to alleviate this problem is to partition the data set into sub-parts. After partitioning the data into 8 different parts we get a fully converged solution with very small errors after only 5 iterations, as compared with previous non-satisfactory results after 100 iterations when not partitioning data.

As this system has a lower noise (less uncertainty) our Taylor approximation will be more accurate, and give us a better result. The opposite is true for higher noise maps.

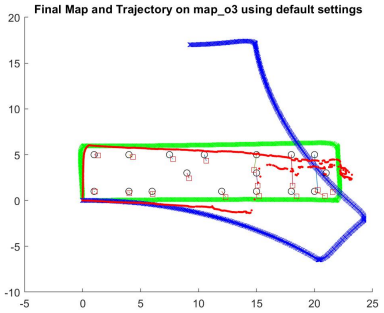


Figure 4: Final map and trajectory for map_o3 when using default settings.

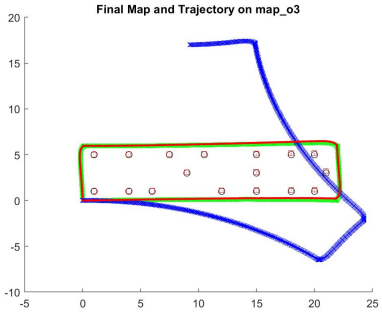


Figure 5: Final map and trajectory for map_o3 after applying improvements.

## 8.2 `map_sym2`

This is a smaller map with four landmarks. This is a relatively easy map to infer as there are few landmarks located symmetrically in a square. We only observe each landmarks once, as we move in a rectangular fashion around the landscape. An exception is the first landmark, which is observed in the beginning and the end of the run.

We do not observe any evidence of local minima as the robot moves on the outside of the landmarks and "wraps" its estimate around the measures.

This map is relatively small, so a too large sensor range could be problematic, if all landmarks were observed from all poses. The information matrix would then be very dense (full with non-zero entries) and thus take longer to solve (using approximation methods).

The noise makes the odometry readings slightly inaccurate, which accumulates over the run, leading the initial estimate to drift away from the true path.

However, in this easy case, where the number of timesteps is relatively small (425) the true path is largely recovered after 20 iterations, as well as the landmark measurements.

When partitioning the data set we get a highly accurate estimate after only 4 iterations.

The starting point here is not important, as the system is symmetric.

The linearization is reasonable when $\bar{\mu}_{i-1}$ is a reasonable estimate for $x_{i-1}$. Otherwise the residual term $(x_{i-1} - \bar{\mu}_{i-1}$ becomes large and the approximation becomes worse. For this map, the estimates are acceptable, since we don't have a large number of time steps. But when $T$ is large and we don't partition the data set, these estimates become increasingly worse, thus making the linear approximation less and less reasonable, as we will see in some of the larger datasets.

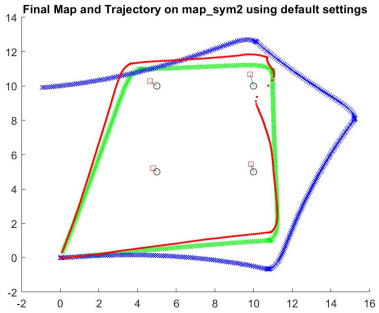


Figure 6: Final map and trajectory for map_sym2 using default settings.

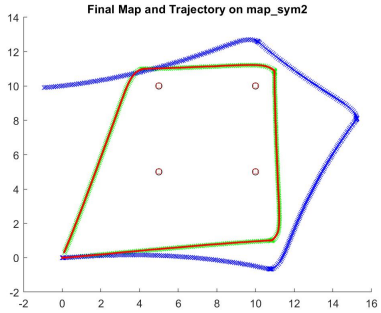


Figure 7: Final map and trajectory for map_sym2 after applying improvements.

## 8.3 `map_sym3`

This is a small rectangular map with five landmarks, evenly distributed over the map. One of the challenges in this map is the fact that the robot path loops over itself for a large portion of the map. This means that we observe the same landmarks at very different points in time, making the inference harder as $\mathbf{\Omega}$ is no longer band-diagonal. Especially challenging with this data set is the long duration of the path. We have 1137 data points, causing excessive error in the approximations as the errors accumulate over time, causing erroneous inference, as seen in figure 8.

In order to alleviate this problem we apply the data partitioning trick, as discussed previously, and receive an excellent result.

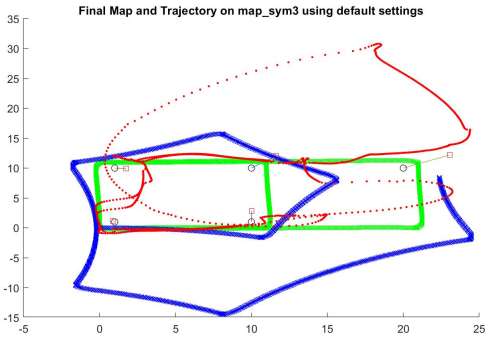


Figure 8: Caption

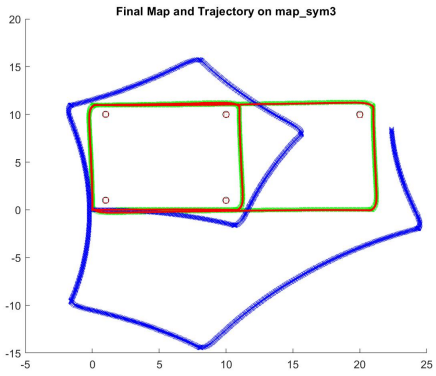


Figure 9: Caption

## 8.4 `map_pent_big_10`

This dataset consists of 1195 poses and poses a new challenge since it contains/allows for outliers. When using default settings the solution diverges. When enlarging $Q$, increasing convergence to $10^{-5}$, allowing for 150 iterations, and partitioning into 4 we get an improved result, but still unsatisfactory due to one outlier. We also partitioned the calculations into 8 parts but without improvement.

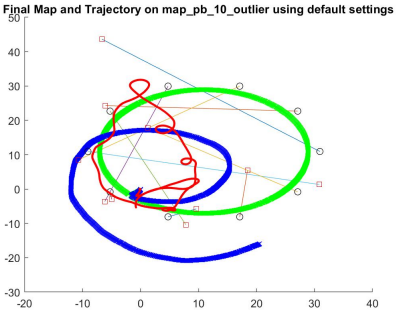


Figure 10: Caption

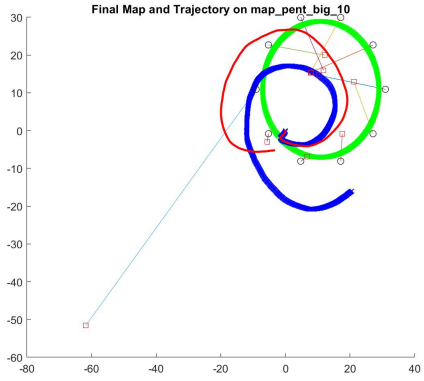


Figure 11: Caption

## 8.5 `map_pent_big_40`

This dataset consists of 239 poses with 40 landmarks located evenly around a circular shape, similar to `map_pent_big_10`, but with more landmarks. The

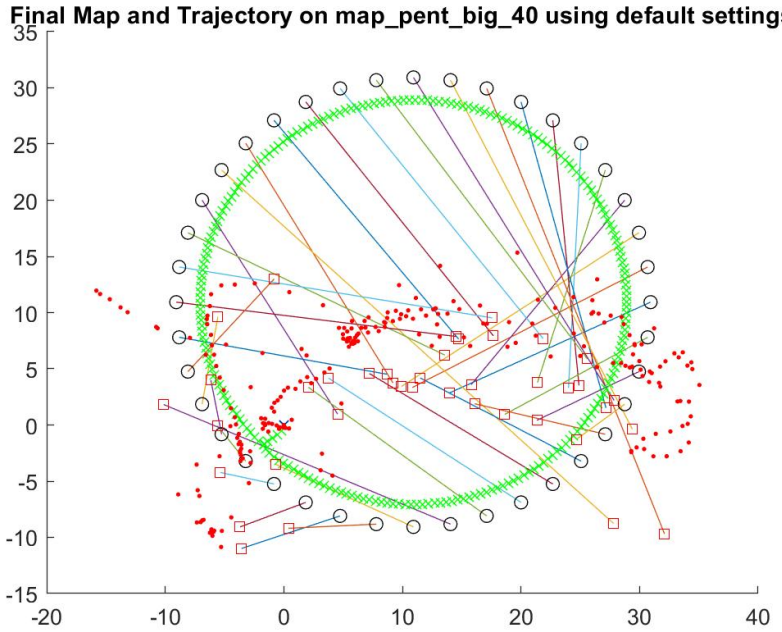


Figure 12: Caption

robot moves on the inside of the circle in less number of timesteps, causing multiple landmarks to be observed simultaneously, which complicates the solution of the system.

As can be seen in figure 12 the results are completely erroneous when using default settings.

# 9 Question 10

There has been a lot of research on the topic of robust methods for Graph-SLAM. For example by introducing a middle-layer in the algorithm, effectively rejecting false-positive loop closures and introduce new artificial loop closures [1] or by optimizing the topology of the graph and identifying false-positive loop closes during optimization [2]. In [3] the authors approach this issue by defining an edge consistency metric to identify outliers and in [4] the authors approach the problem by allowing for richer error models, allowing the probability of a false-positive loop closures to be explicitly modeled.

However, none of these are particularly straight forward to implement and were not applied to the problem at hand due to time constraint.

## 10 Question 11

For the last two data sets we identify outliers by measuring the distance between the current landmark coordinate with the previous and the next one. If both distances are abnormal (larger than some threshold based on mean distance between landmarks in complete dataset) we replace the outlier value with the midpoint coordinate between the previous and next landmark. This, in

## References


[1] Xie, L., Wang, S., Markham, A., & Trigoni, A. (2017). GraphTinker: Outlier rejection and inlier injection for pose graph SLAM. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 6777-6784.

[2] Sünderhauf, & N., Prozel, P. (2012). Towards a Robust Back-End for Pose Graph SLAM. 2012 IEEE Intl. Conf. on Robotics and and Automation (ICRA), 6224-709.

[3] Knuth, J., & Barooah, P. (2013). Outlier rejection for pose graph optimization. In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS).

[4] Olson, E., & Agarwal, P. (2013). Inference on networks of mixtures for robust robot mapping. The International Journal of Robotics Research, 32(7), 826-840.